LinuxDay 2008

WPKG

http://wpkg.org/

Pordenone, ITIS J. F. Kennedy, 25/10/2008

Marco Gaiarin <gaio@linux.it>

Il problema

La legge sulla privacy e il buon senso impongono un aggiornamento di tutto il software sui sistemi; inoltre è molto utile poter installare programmi e impostazioni a nostro piacimento.

- in AD esisterebbe il framework GPO, ma Samba non lo supporta, e poi funziona solo con installer MSI...
- non possiamo usare le policy normali (NT) per tutto...
- non possiamo usare i logon script, visto che vengono eseguiti con i permessi dell'utente...

Il paradosso dell'amministratore di rete *libero*...

La soluzione

Di cosa abbiamo bisogno realmente? Di *qualcosa* che:

- venga eseguito con permessi sufficientemente elevati nelle varie macchine client;
- permetta una verifica delle operazioni, non possiamo fidarci dei client...
- permetta non solo di installare/applicare, ma anche di disinstallare e aggiornare
- sia il più efficiente possibile, ovvero non compia operazioni se non strettamente necessario.

Tutto questo è WPKG!!!

Considerazioni generali

- Non va visto come un sistema per *risparmiare tempo*: almeno all'inizio il tempo sprecato per preparare le *ricette* e comprendere il sistema è una vera enormità...
- Va inoltre ricordato che si tratta di una necessità data da un obbligo di legge, indi il costo c'è e non è eludibile (non è un giochino, anche se è divertente ;).
- I conti vanno fatti nel medio/lungo periodo, e confrontandoli con l'alternativa di girarsi tutti i PC (fisicamente, con VNC o psexec).
- Occorre anche vedere il resto del sistema...

Cos'è WPKG

Un sistema PUSH/PULL di software e configurazioni da un server a n client

- sostituisce decentemente psexec e le GPO (per la parte di deploy, of course)
- eseguibile tramite schtasks, logon scripts, GPO, ...
- ha il suo client (WPKGClient)
- completamente server-side, è difficile *rompere* qualcosa
- Strutturalmente molto semplice, scriptabile, estendibile a piacere: totale libertà all'amministratore!

Cosa NON è WPKG

- WPKG non è un sistema di provisioning (sysprep, RIS/WDS, unattended, ...), anche se può essere usato per scopi simili.
- WPKG non è un sistema di generazione e gestione dell'inventario HW/SW, anche se ovviamente tramite WPKG si può installare e configurare questi sistemi (vedi OCSInventory-ng)
- WPKG non è e non potrà mai essere un sistema point'n'click, anche se ovviamente qualche semplificazione può essere fatta...

Come funziona WPKG

- Tramite uno dei metodi elencati (schtasks, logon script, GPO, WPKGClient) viene eseguito WPKG, uno script che risiede server-side.
- Lo script opera client-side utilizzando un database (XML) sul server, tramite semplice share di rete.
- Il database contiene un elenco di istruzioni di installazione (**package** o *ricette*), un elenco di **profili** (composti da una o più ricette) e un elenco di afferenza dei vari **host** ai vari profili.
- Lo script salva localmente un database (XML) delle ricette installate (lo stato del sistema client).

Server WPKG

Visto che il database di WPKG è un insieme di file XML, il server non sarà altro che uno share samba:

```
[wpkg]
        comment = WPKG Automated Software Deploying System
        path = /etc/samba/wpkg
        guest ok = Yes
        browseable = No
        share modes = No
        writable = No
        write list = root @ced
```

Notare che si tratta di uno share *tecnico*, e che il guest access è abilitato: ho infatti scelto di installare tramite WPKG solo software libero/gratuito.

Un problema se occorrono le chiavi delle licenze...

Struttura della cartella WPKG

- `wpkg.js`: lo script WSH cuore del sistema http://it.wikipedia.org/wiki/Windows_Script_Host
- `packages.xml`, `packages/`: elenco di ricette di installazione e configurazione
- `profiles.xml`, `profiles/`: profili, ovvero insiemi di di ricette di installazione e configurazione
- `hosts.xml`, `hosts/`: assegnazione delle varie macchine ad un profilo
- `tools/`: utility utilizzate dalle ricette e potenzialmente non installate negli hosts (decompressori, ...)

Profili

I profili sono un elenco di ricette associabili ad un host. C'è anche la possibilità di far dipendere i profili da altri profili.

```
<profile id="segreterie">
        <depends profile-id="default" />
        <package package-id="psqlodbc" />
        <package package-id="centura" />
        <package package-id="centura-settings" />
</profile>
```

C'è solo la possibilità di includere dipendenze, e non di sottrarre dipendenze e/o pacchetti.

Profilare le proprie esigenze è la prima cosa da fare...

Host

Tramite questi file associo ad un host un particolare profilo, o più di uno.

Posso utilizzare delle wildcard/espressioni regolari.

```
<host name=".+" profile-id="default" />
<host name="ced.+" profile-id="ced" />
<host name="ted" profile-id="segreterie" />
<host name="maso" profile-id="segreterie" />
<host name="igor" profile-id="segreterie" >
   <profile id="paghe" >
</host>
```

L'host è, ovviamente, il nome della macchina windows, case insensitive come si usa in quel mondo.

Ricette

Una ricetta ha una struttura del tipo:

```
<package
  id="putty"
  name="Putty"
  revision="59"
  priority="50"
  reboot="false">

  <check type="uninstall" condition="exists" path="PuTTY version 0.59" />

  <install cmd='%SOFTWARE%\WPKG\putty-0.59-installer.exe /sp- /silent' />
  <upgrade cmd='%SOFTWARE%\WPKG\putty-0.59-installer.exe /sp- /silent' />
  <remove cmd='"%ProgramFiles%\PuTTY\unins000.exe" /sp- /silent /norestart' />
</package>
```

per facilitare la gestione è fortemente consigliato parametrizzare le ricette con delle variabili d'ambiente, standard di Windows o definite da noi.

Ricette: parametri di base

- id: l'identificativo del pacchetto
- name: il nome, viene usato nei log
- revision: informazioni di versione: se la versione cambia (cresce) scatta l'operazione di upgrade
- priority: priorità di installazione, in caso di dipendenze fa fede la priorità e non la catena delle dipendenze (vedi poi)
- reboot: se la corretta installazione o aggiornamento del pacchetto necessita di un riavvio, e quando.

Ricette: dipendenze

```
<package
        id="thunderbird-settings"
        name="Mozilla Thunderbird configuration"
        revision="2"
        reboot="false"
        priority="5">
        <depends package-id="thunderbird"/>
```

É possibile esplicitare una dipendenza di un pacchetto da un altro: installando un pacchetto verranno automaticamente installate tutte le dipendenze.

Come già detto, questo non influisce sull'ordine di installazione, e la dipendenza è esclusivamente funzionale all'installazione, non all'uso.

Ricette: verifiche

All'interno di una ricetta è possibile definire delle condizioni:

- registry: verifica che una particolare chiave di registro esista o abbia un certo valore
- file: verifica che un particolare file esista e abbia certe proprietà
- uninstall: verifica che il programma in questione sia elencato nella lista dei programmi installati

Le condizioni possono essere semplici e combinate, ovviamente con la logica booleana (and, or, not).

Ricette: logica delle verifiche

Le condizioni di verifica vengono valutate:

- prima dell'installazione: se la condizione è vera si suppone che il software sia già installato, il sistema segnala un warning e procede oltre.
- dopo l'installazione: se la condizione è falsa, l'installazione non è andata a buon fine, e il sistema semplicemente riproverà alla successiva chiamata

In questa maniera non viene forzata una installazione se è già presente il software, e l'installazione viene tentata fino al successo della condizione.

Ricette: logica delle verifiche/2

- Le condizioni vengono verificate comunque ad ogni esecuzione, questo impedisce la disinstallazione manuale
- In aggiornamento vengono comunque valutate le verifiche: se è falsa all'inizio, scatta l'installazione e non l'aggiornamento
- In rimozione, ovviamente, le condizioni vengono verificate in modo duale

Ricette: execute once/always

```
   execute="once">

   <install cmd="regedit /s %WPKGROOT%\packages\thunderbird.reg" />
   <upgrade cmd="regedit /s %WPKGROOT%\packages\thunderbird.reg" />

</package>
```

Tramite il parametro execute si modifica la logica:

- **once**: la ricetta viene eseguita una volta sola senza condizioni, e rieseguita se la revision cresce
- **always**: la ricetta viene eseguita sempre

In entrambi i casi non vengono valutate le condizioni; questa modalità è comoda per le impostazioni.

Ricette: comandi

I comandi da eseguire sono normali comandi, batch, ...

- per ogni tipologia posso inserire istanze a piacere, verranno eseguite e valutate in ordine
- se ovviamente il task è particolarmente complesso, è bene costruire un file batch ed eseguire quello
- i comandi devono essere costruiti per emettere il minimo output e fare zero domande: le risposte non possono essere lasciate all'utente!!!
- per ogni istanza posso condizionare il reboot ad un particolare stato di uscita

Ricette: azione install

- Viene eseguita solo per la prima installazione (aggiunta di una ricetta a un profilo, prima installazione della macchina, ...) o nel caso di non-verifica delle condizioni
- Le condizioni vengono verificate prima e dopo l'esecuzione delle azioni
- Lo stato di uscita viene valutato

Ricette: azioni upgrade

- viene eseguita solo se cambia (cresce) la revisione della ricetta
- le condizioni vengono controllate, se non sono verificate inizialmente scatta l'operazione di install
- lo stato di uscita viene valutato

Ricette: azione remove

- viene eseguita quando si rimuove una ricetta dal profilo corrente dell'host
- le condizioni vengono verificate (ovviamente dualmente rispetto a install) sia prima che dopo
- lo stato di uscita viene valutato
- **ATTENZIONE** che per l'azione di remove viene utilizzata la voce sul database locale e non quella sul database server side (formalmente corretto)
- per questo in ogni caso viene eseguita l'azione upgrade prima della remove se necessario...

Ricette: reboot

Le azioni di reboot possono essere controllate a livello globale (/noreboot), a livello di ricetta e a livello di stato di uscita (reboot flag).

Il valore del reboot flag può essere, oltre che true/false, anche delayed (fine della ricetta) o postponed (fine dell'esecuzione di WPKG).

```
<upgrade cmd='msiexec /q /x{AC76BA86-7AD7-1040-7B44-A81200000003}
  REBOOT=ReallySuppress' >
     <exit code="1605" />
     <exit code="3010" reboot="postponed" />
  </upgrade>
```

Ricette: note finali

- La congruenza tra database locale e sistema reale passa per le check condition, è essenziale scrivere buone condizioni di verifica...
- A parte eseguire wpkg.js con le opzioni di debug, ogni cosa viene segnalata nel log di sistema (EventViewer)
- Se si vuole eseguire un reset dello stato di WPKG, si può benissimo eliminare il database locale: le check condition impediranno di reinstallare tutto.
- Molti installer si comportano meglio se installati ed aggiornati con lo stesso utente (consigliato: SYSTEM)

Scrivere ricette

- molti installer fanno schifo, occorre studiare, pianificare e testare bene ogni ricetta... alcuni problemi sono difficili da scovare...
- gli installer MSI sono i più facili: se sono bacati si possono sistemare con ORCA, l'unica avvertenza è di non eliminare la copia vecchia in aggiornamento...
- Siti consigliati sono http://wpkg.org/ , http://unattended.sf.org/ , http://appdeploy.com/ e http://www.ss64.com/ .
- Su google le chiavi di ricerca sono *unattended* oppure *silent installation* oppure *deploy*.

Gestione

- WPKGWeb (http://wpkg.linuxkidd.com/)
  Interfaccia web di gestione completa, repository di ricette (in stallo)
- http://joliot-curie.ens-lyon.fr/wpkg/
  Interfaccia semplificata, definisco le ricette a mano, qui solo definisco profili e li assegno
- Una buona politica di definizione delle ricette e CVS/RSync/...
- Una buona politica dello share dove tenere gli installer dei programmi

WPKGClient

Si tratta del metodo preferenziale per eseguire WPKG.
Vantaggi:

- Posso scegliere con che utente eseguirlo e con che utente accedere agli share
- Posso definire delle variabili d'ambiente
- Permette di eseguire un pre e un post task
- Posso configurare il servizio tramite se stesso (utilizza un file XML per la configurazione)
- Permette di eseguire WPKG al boot o allo shutdown, e di ritardare il logon/logoff con un messaggio

Le mie linee guida...

Alcune scelte di fondo:

- variabili d'ambiente
- bootstrap del sistema
- pacchetti -settings
- profili ed host

Inoltre come già accennato la scelta è quella di non usare WPKG per sw con licenza, quindi viene usato per l'installazione l'utente SYSTEM (default) e l'accesso guest al server WPKG e al repository del software.

Linee: variabili d'ambiente

- `%WPKGROOT%`: variabile suggerita, punta allo share (*server*) WPKG
- `%SOFTWARE%`: variabile suggerita, punta al repository del software
- `%VPSERVER%`: server antivirus
- `%LOCALNET%`: identificativo rete locale

L'obiettivo è quello di fare un set di ricette uniche per tutte le installazioni, e parametrizzarle.

Il problema dei portatili che girano tra più server WPKG: share con password, o pre-task con verifica.

Linee: bootstrap

Come già accennato molti installer non si comportano bene se installati con un utente e aggiornati con un altro.

Il sistema di bootstrap è pensato per forzare l'utilizzo dell'utente SYSTEM anche per la prima installazione, ma contemporaneamente preservando l'interattività della prima installazione.

Linee: pacchetti -settings

Se un programma ha bisogno di alcune configurazioni post-installazione (che possono cambiare), viene creato un ulteriore pacchetto con lo stesso nome e il suffisso -settings con queste impostazioni.

In questa maniera posso aggiornare le impostazioni senza essere costretto a reinstallare il programma.

Ovviamente le impostazioni dipendono dal programma onde evitare di applicare impostazioni che non verranno usate.

Linee: profili ed host

- viene creato un profilo *base* con il sw minimo necessario
- vengono creati due profili, *default* e *portatili*, figli di *base*
- vengono creati altri profili in base all'uso (ced, segreterie, assistiti, media, diretta, ...) ognuno con le loro specificità, tutti figli di default o di un figlio di default.

Esempi

- esempio di ricetta semplice: putty
- esempio di ricetta mediamente complicata: savce
- esempio di ricetta complicata: informix
- esempio di ricetta molto complicata: oracle
- ...
- ...